# Approximation and Computational Complexity of Some Hammock Variations of the Poset Cover Problem

Ivy Ordanel

Proceso L. Fernandez Jr

Richelle Ann B. Juayong

Henry N. Adorna

**Philippine Journal of Science**
149 (1): 201-211, March 2020
ISSN 0031 - 7683
Date Received: 22 Oct 2019

# Approximation and Computational Complexity of Some Hammock Variations of the Poset Cover Problem

**Ivy D. Ordanel**[1,2]*, **Proceso L. Fernandez Jr.**[2],
**Richelle Ann B. Juayong**[1], and **Henry N. Adorna**[1]

[1]Department of Computer Science,
University of the Philippines Diliman, Quezon City 1101 Philippines
[2]Department of Information Systems and Computer Science,
Ateneo De Manila University, Quezon City 1108 Philippines

**The Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem is a variation of the Poset Cover Problem with the same input – set $\{L_1, L_2, \dots, L_m\}$ of linear orders over the set $\{1,2,\dots,n\}$, but the solution is restricted to a set of simple hammock($\underbrace{2,2,\dots,2}_{k}$) posets. The problem is NP-Hard when $k \geq 3$ but is in $P$ when $k = 1$. The computational complexity of the problem when $k = 2$ is not yet known. In this paper, we determine the approximation complexity of the cases that have been shown to be NP-Hard. We show that the Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem is in $APX$ and, in particular, $(1 + \frac{1}{2^k})$-approximable, for $k \geq 3$. On the other hand, we also explore the computational complexity for the case where $k = 2$ [Hammock(2,2)-Poset Cover Problem]. We show that it is in $P$ when the transposition graph of the input set of linear orders is rectangular.**

Keywords: algorithm, approximation, complexity, partial order, poset

## INTRODUCTION

Sequential data such as logs may contain a wealth of information that can be mined to discover knowledge about the objects in the data or about the system that generates the data. One interesting aspect of the objects in the data is their dependencies or ordering. The task of determining such is more commonly referred to as mining posets – a data mining task that is relevant in bioinformatics, process model mining, web mining, network management and intrusion detection, and preference-based service (Pei *et al.* 2006).

A (strict) partially ordered set or poset $P = (V, <_P)$, is a pair consisting of a finite set $V$ and a partial ordering on it that is defined by an irreflexive, antisymmetric and transitive binary relation $<_P$. Poset $P$ becomes a totally ordered set or a linear order when every pair of elements in $V$ are related, *i.e.* $(u, v) \in <_P$ or $(v, u) \in <_P$ for every distinct $u, v \in V$. Moreover, a linear order, say $L = (V, <_L)$ is a linear extension of a poset $P$ if $<_P \subseteq <_L$. We denote the set of all linear extensions of poset $P$ as $\mathcal{L}(P)$. For example, Figure 1c shows the set of all linear extensions of the poset defined in Figure 1a.

*Corresponding Author: ivyordanel@gmail.com

Using these mathematical structures, Heath and Nema (2013) abstracted the said data mining task as the Poset Cover Problem, where they represented the input sequences as linear orders and the solution is one or more posets that explain, through their linear extensions, the ordering in the linear orders. This is formally defined as follows:

**Poset Cover Problem**
Instance: A set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of linear orders over the set $V = \{1,2,3,\ldots,n\}$.

Solution: A set $P^* = \{P_1, P_2, \ldots, P_q\}$ of posets where $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$ and $q$ is minimum.

Heath and Nema (2013) showed that the decision version of the Poset Cover Problem is NP-Complete. Fernandez *et al.* (2013) also studied the hardness of the problem for different classes of posets with respect to their Hasse diagram. Every poset corresponds to a directed acyclic graph (DAG) $G = (V, E)$ with vertex set $V$ and edge set $E = \{(u, v) \in <_P\}$. The Hasse diagram of poset $P = (V, <_P)$, denoted as H(P), is the transitive reduction of the DAG $G$. Figure 1b shows the Hasse diagram of poset $P$ defined in Figure 1a. Classes of posets according to their Hasse diagram include leveled posets and hammock posets. A leveled poset $P = (V, <_P)$ is a poset where the set $V$ can be partitioned into different levels $V_1, V_2, \ldots, V_k$ such that for $u \in V_i$ and $v \in V_j$, $u <_P v$ if and only if $i < j$. Leveled posets could be used as combinatorial models for workflow diagrams in process model mining since workflow diagrams could have also a series of different stages or levels. On the other hand, a hammock poset is a leveled poset where $|V_1| = |V_k| = 1$ and either $|V_i| = 1$ or $|V_{i+1}| = 1$ for $2 \leq i \leq k-1$. A non-singleton $V_i$ in a hammock poset is called a hammock. Figure 1b shows the Hasse diagram of a hammock-poset with hammocks {1,9} and {2,4,5}. The sequence of the sizes of the hammocks is used to name the class of hammock posets. The hammock$(a_1, a_2, \ldots, a_n)$ is a class of hammock posets where $a_i$ is the size of the $i$th hammock. Hence, the poset in Figure 1 is a hammock(2,3) poset. Hammock posets are also used as a combinatorial model to represent mixed-initiative dialog systems (Fernandez *et al.* 2013).

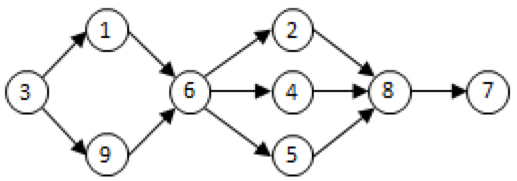
| | | |
|---|---|---|
| P=(V,<$_P$) where<br>V={1,2,3,4,5,6,7,8,9} and<br><$_P$={(3,1),(3,9),(3,6),(3,2),(3,4),<br>(3,5),(3,8),(3,7),(1,6),(1,2),<br>(1,4),(1,5),(1,8),(1,7),(9,6),<br>(9,2),(9,4),(9,5),(9,8),(9,7),<br>(6,2),(6,4),(6,5),(6,8),(6,7),<br>(2,8),(2,7),(4,8),(4,7),(5,8),<br>(5,7),(8,7) } |  | $\mathcal{L}$(P) = {(3,1,9,6,2,4,5,8,7),<br>(3,1,9,6,2,5,4,8,7),<br>(3,1,9,6,5,2,4,8,7),<br>(3,1,9,6,5,4,2,8,7),<br>(3,1,9,6,4,2,5,8,7),<br>(3,1,9,6,4,5,2,8,7),<br>(3,9,1,6,2,4,5,8,7),<br>(3,9,1,6,2,5,4,8,7),<br>(3,9,1,6,5,2,4,8,7),<br>(3,9,1,6,5,4,2,8,7),<br>(3,9,1,6,4,2,5,8,7),<br>(3,9,1,6,4,5,2,8,7)} |
| (a) P=(V,<$_P$) | (b) H(P) | (c) $\mathcal{L}$(P) |

**Figure 1.** Poset P and its Hasse diagram and linear extensions.

In our study, we only consider hammock($\underbrace{2,2,\ldots,2}_{k}$) posets that have $k$ hammocks of size 2. Note that these posets have exactly $2^k$ linear extensions; hence, we can say that they are a simple class of posets. It is also then interesting to determine the hardness of the problem in this simple class of posets. We can then define the variant of the problem for this as follows:

**Hammock($\underbrace{2,2,\ldots,2}_{k}$)-Poset Cover Problem**
Instance: A set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of linear orders over the set $V = \{1,2,3,\ldots,n\}$.

Solution: A set $P^* = \{P_1, P_2, \ldots, P_q\}$ of hammock($\underbrace{2,2,\ldots,2}_{k}$) posets where $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$ and $q$ is minimum.

The decision version of the Hammock($\underbrace{2,2,\ldots,2}_{k}$)-Poset Cover Problem is NP-Complete for $k \geq 3$ and in $P$ for $k = 1$ (Fernandez *et al.* 2013). The computational complexity of the problem when $k = 2$ is not yet known. In our previous study (Ordanel and Adorna 2018), we presented a solution for $k = 2$; however, the solution runs in exponential time in the worst case.

For problems that have been shown to be NP-hard, one common approach is approximation. In approximation, we want to find an algorithm that produces a solution that is not necessarily optimal but with a cost that is guaranteed to be within a factor of the optimal solution. With this, another measure of the complexity of hard optimization problem is to determine how hard it is to approximate the problem. This is the first part of our results; we determine the approximation complexity of the problem that has been shown to be NP-hard which is the Hammock($\underbrace{2,2,\ldots,2}_{k}$)-Poset Cover Problem for $k \geq 3$. We have rewritten and extended our results from Ordanel et al. (2017) for this.

On the other hand, in determining the computational complexity for the case where $k = 2$ or the Hammock(2,2)-Poset Cover Problem, we first explore some of its simpler cases – a common approach in dealing with hard problems. For instance, the well-known Hamilton Path Problem is known to be NP-hard. Itai et al. (1982) studied the case where the input graph to the problem is a rectangular graph and the problem is in $P$ for this simple case. In a similar way, for the Hammock(2,2)-Poset Cover problem, we study and determine the computational complexity of the problem when the transposition graph of the input set of linear orders is a rectangular graph.

**Definitions**
In this section, we establish the other terminologies and notations used in the discussion of results.

**Grid Graph (Itai *et al.* 1982)**
Let $G^\infty$ be the infinite graph whose vertex set consists of all points in a plane with integer coordinates and in which two vertices are connected if and only if the (Euclidean) distance between them is equal to 1. A **grid graph** is a finite, node-induced subgraph of $G^\infty$.

**Rectangular Graph (Itai *et al.* 1982)**
Let $R(m,n)$ be the grid graph whose vertex set is $V(R(m,n)) = \{(v_x, v_y): 1 \leq v_x \leq m \text{ and } 1 \leq v_y \leq n\}$. A rectangular graph is a grid graph which, for some $m$ and $n$, is isomorphic to $R(m,n)$.

**Adjacent Transposition Graph**
We say that two linear orders differ by a single adjacent transposition if one can be transformed to the other by interchanging or swapping two adjacent elements. The adjacent transposition graph $G = (V, E)$ is an undirected graph where $V$ is a set of linear orders over a given base set and $E = \{\{L_1, L_2\} | L_1$ and $L_2$ differ by a single adjacent tranposition$\}$. Given a set of linear orders $\Upsilon$, we denote the adjacent transposition graph of $G = (\Upsilon, E)$ as $\mathcal{G}(\Upsilon)$.

**Rectangular Hammock(2,2)-Poset Cover Problem**
Instance: A set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of linear orders over the set $V = \{1,2,3,\ldots,n\}$ where $\mathcal{G}(\Upsilon)$ is a rectangular graph

Solution: A set $P^* = \{P_1, P_2, \ldots, P_q\}$ of hammock(2,2) posets where $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$ and $q$ is minimum

**Cover**
Given a set of linear orders $\Upsilon$ and poset $P$, we say that $P$ covers $\Upsilon$ if and only if $\mathcal{L}(P) = \Upsilon$. Given a set of linear orders $\Upsilon$ and a set of posets $P^*$, we say that $P^*$ covers $\Upsilon$ if and only if $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$.

**NP Optimization Problem (Ausiello and Paschos 2005; Crescenzi 1997)**
An NP optimization problem, NPO, $\pi$ is defined as a four-tuple $(\mathcal{I}, Sol, m, goal)$ where:

- $\mathcal{I}$ is the set of instances of $\pi$ and it can be recognized in polynomial time;
- given $x \in \mathcal{I}$, $Sol(x)$ denotes the set of feasible solutions of $x$; for any $y \in Sol(x)$, $|y|$ is polynomial in $|x|$; given any $x$ and $y$ polynomial in $|x|$, one can decide in polynomial time if $y \in Sol(x)$;
- given $x \in \mathcal{I}$ and $y \in Sol(x)$, $m(x,y)$ denotes the value of $y$ and can be computed in polynomial time; and
- $goal \in \{min, max\}$ indicates the type of optimization problem.

**k-Set Cover Problem**
- $\mathcal{I}$ : Finite set $S$ and a collection $C$ of subsets of $S$ where $|c| \leq k$, for $c \in C$;
- $Sol$: $C' \subseteq C$ such that every element in $S$ belongs to at least one member of $C'$;
- $m$: $|C'|$; and
- $goal$: $min$

The decision version of the $k$-Set Cover Problem is NP-Hard for $k \geq 3$ (Ausiello *et al.* 1999).

**Approximation Ratio**
Let $\pi$ be an NPO problem. Given an instance $x$ and a feasible solution $y$, we define the approximation or performance ratio $\rho$ of $y$ with respect to $x$ as the ratio between $m(x,y)$ and the optimal solution $opt(x)$, *i.e.*:

$$\rho(x,y) = max\left\{\frac{m(x,y)}{opt(x)}, \frac{opt(x)}{m(x,y)}\right\}$$

**Strict Reduction** (Ausiello and Paschos 2005; Crescenzi 1997)

Let $\pi_1$ and $\pi_2$ be two NPO minimization problems. Then, we say that $\pi_1$ is strictly reducible to $\pi_2$, denoted by $\pi_1 \leq_S \pi_2$, if there exist two polynomial time computable functions $f, g$ that satisfy the following properties:

- $f\colon \mathcal{I}_{\pi_1} \rightarrow \mathcal{I}_{\pi_2}$ , *i.e.* $f$ maps instances of $\pi_1$ into instances of $\pi_2$;
- $g\colon \mathcal{I}_{\pi_1} \times Sol_{\pi_2} \rightarrow Sol_{\pi_1}$ , *i.e.* $g$ maps back solutions of $\pi_2$ into solutions of $\pi_1$; and
- for all $x \in \mathcal{I}_{\pi_1}$ and for all $y \in Sol_{\pi_2}(f(x)), \rho_{\pi_2}(f(x),y) \geq \rho_{\pi_1}(x, g(x,y))$.

## RESULTS AND DISCUSSION

**Approximation Complexity of Hammock($\underbrace{2, 2, \ldots, 2}_{k}$)-Poset Cover Problem**
There is already a result in the literature where, given a set of linear orders $\Upsilon$ and an integer $k$, we can efficiently get all the possible hammock posets with $k$ hammocks of size 2 whose union cover $\Upsilon$. This is stated as follows.

***Theorem 1*** (Ordanel and Adorna 2018)***:*** Given a set of linear orders $\Upsilon = \{L_1, L_2, \dots, L_m\}$ over the set $V = \{1,2,\dots,n\}$ and an integer $k > 0$, we can determine the set of posets $P^* = \{\, P$ is hammock($\underbrace{2,2,\dots,2}_{k}$) poset and $\mathcal{L}(P) \subseteq \Upsilon \,\}$ such that $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$ in $O(kn^2\, m^2)$-time.

We denote the algorithm that determines $P^*$ as $GetAllHammock2kPosets$. As an example, given the set of linear orders $\Upsilon$ in Figure 2a and $k = 2$, $GetAllHammock2kPosets$ determines the set of Hammock(2,2)-Posets $P^* = \{P_1, P_2, P_3\}$ whose Hasse diagrams are illustrated in Figure 2b.

Note that $P^*$ is not necessarily optimal – a subset of $P^*$ could be enough to exactly cover $\Upsilon$. For instance, in the example given in Figure 2, we can see that the subset $\{P_1, P_3\}$ is the optimal solution. To approximate then the Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem, we can approximate the minimum subset of $P^*$ that still cover $\Upsilon$. To do this, note that each $P_i$ in $P^*$ corresponds to its set of linear extensions $\mathcal{L}(P_i) \subseteq \Upsilon$. From this, we can then reduce the problem to a variant of the Set Cover Problem. We establish the reduction with the following results.

$L_1 = (1,4,3,2,5,6,8,7,9)$

$L_2 = (1,3,4,2,5,6,8,7,9)$

$L_3 = (1,4,3,2,5,6,7,8,9)$

$L_4 = (1,3,4,2,5,6,7,8,9)$

$L_5 = (1,4,3,2,5,7,6,8,9)$

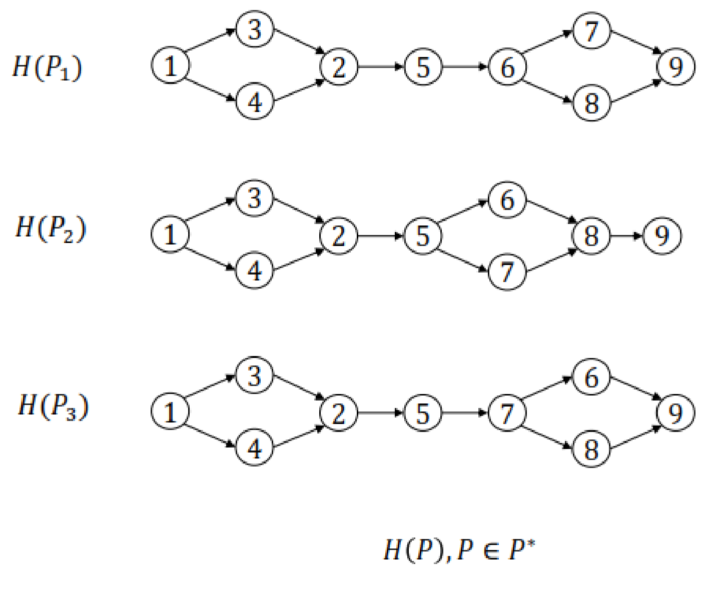
$L_6 = (1,3,4,2,5,7,6,8,9)$

$L_7 = (1,4,3,2,5,7,8,6,9)$

$L_8 = (1,3,4,2,5,7,8,6,9)$

$\Upsilon$

$(a)$



$(b)$

**Figure 2.** Example of GetAllHammock2kPoset Algorithm.

***Lemma 2****:* Given a poset $P = (V, <_P)$, if $P$ is a hammock($a_1, a_2, \dots, a_y$), then $|\mathcal{L}(P)| = a_1!\ a_2! \dots a_y!$.

***Proof:*** Let $n = |V|$. To form a linear extension $l$ of length $n$, we first place all the link vertices in order while leaving respective spaces for the hammock vertices. Then, place elements of hammock $a_1$ in the space allotted for $a_1$. There are $a_1!$ possible arrangement of the elements of hammock $a_1$. Do this for hammock $a_2$ to $a_y$. Hence, by the Multiplication Principle, $|\mathcal{L}(P)| = a_1!\ a_2! \dots a_y!$.

***Theorem 3:*** Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem $\leq_S$ $k'$-Set Cover Problem, where $k' = 2^k$.

***Proof***: Let $\pi_1$ be the Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem and $\pi_2$ be the $k'$-Set Cover Problem where $k' = 2^k$.

Let $\Upsilon \in \mathcal{I}_{\pi_1}$.

We define $f$ to be the following:

```
1. $P^* \leftarrow$ GetAllHammock2kPosets($\Upsilon, k$)
2. $\tau \leftarrow \{\mathcal{L}(P) | P \in P^*\}$
3. Return $(\Upsilon, \tau)$
```

From Theorem 1, $P^* = \{P | P$ is hammock($\underbrace{2,2,\dots,2}_{k}$) poset and $\mathcal{L}(P) \subseteq \Upsilon\}$ and $\bigcup_{P \in P^*} \mathcal{L}(P) = \Upsilon$. Hence, $\tau = \{\mathcal{L}(P) | P \in P^*\}$ is a collection of subsets of $\Upsilon$ and $\bigcup_{t \in \tau} t = \Upsilon$. Moreover, from Lemma 2 $|\mathcal{L}(P)| = 2^k$. Hence, $|t| = 2^k = k'$ for each $t \in \tau$. Clearly, $f(\Upsilon) = (\Upsilon, \tau) \in \mathcal{I}_{\pi_2}$. From Theorem 1, we can determine $P^*$ using $GetAllHammock2kPosets$ in polynomial-time. $\mathcal{L}(P)$ can be determined in two ways. First, since $GetAllHammock2kPosets$ builds hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset $P$ by combining its linear extensions, then $\mathcal{L}(P)$ can be stored efficiently as a property of a poset $P$. Second, given a poset $P$ we can also determine $\mathcal{L}(P)$ using the algorithm of Pruesse and Ruskey (1994). Thus, we can say that $f$ is a polynomial computable function.

The problem of determining a minimum Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset that covers $\Upsilon$ can then be translated to the problem of determining the minimum set cover of $\Upsilon$ from subsets in $\tau$. If $x \in \mathcal{I}_{\pi_1}$, $y \in Sol_{\pi_2}$ for $f(x)$, we can define polynomial computable function $g(x, y) = \{P \in P^* | \mathcal{L}(P) \in y\}$.

From these premises, we can clearly infer that the same approximation ratio applies, *i.e.* $\rho_{\pi_2}(f(x), y) = \rho_{\pi_1}(x, g(x, y))$. The shown reduction is hence a strict reduction. ■

***Corollary 4:*** Given an integer $k \geq 3$, Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem is in $APX$. Specifically, it is $(1 + \frac{1}{2^k})$-approximable.

***Proof:*** The $k'$-Set Cover Problem is in $APX$ for $k' \geq 3$ (Halldorsson 1996). Strict reduction preserves the membership in $APX$ (Crescenzi 1997; Ausiello and Paschos 2005). Hence, Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem where $k \geq 3$ is also in $APX$.

In the literature, the best approximation ratio to our knowledge for the $k'$-Set Cover Problem is the $(1 + \frac{1}{k'})$-approximation algorithm of Essa *et al.* (2016) for $k' \geq 6$. Since, $k \geq 3$, then $k' \geq 8$; hence, we can use the algorithm. From the proof of Theorem 3, the same approximation ratio also applies to Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem. Hence, Hammock($\underbrace{2,2,\dots,2}_{k}$)-Poset Cover Problem is $(1 + \frac{1}{2^k})$-approximable, for $k \geq 3$. ■

**Computational Complexity of Rectangular Hammock(2,2)-Poset Cover Problem**

In this section, we present a polynomial-time solution for the Rectangular Hammock(2,2)-Poset Cover Problem. From Theorem 1, given $\Upsilon$ and $k = 2$, we can get efficiently using $GetAllHammock2kPosets$ Algorithm the set $I^* = \{P$ is a hammock(2,2) poset and $\mathcal{L}(P) \subseteq \Upsilon\}$ where $\bigcup_{P_i \in I^*} \mathcal{L}(P_i) = \Upsilon$. What remains then is to determine the minimum subset of $I^*$ that covers $\Upsilon$. Remember that for Rectangular Hammock(2,2)-Poset Cover Problem we have an additional assumption, *i.e.* $\mathcal{G}(\Upsilon)$ is rectangular. Let us then first characterize the transposition graph $\mathcal{G}(\Upsilon)$ with the help of the following previous result.

***Lemma 5*** (Ordanel and Adorna 2018)***:*** Given two distinct hammock(2,2) posets $P_1 = (V, <_{P_1})$ and $P_2 = (V, <_{P_2})$ and let $\mathcal{G}(\mathcal{L}(P_1))$ and $\mathcal{G}(\mathcal{L}(P_2))$ be their respective adjacent transposition graph. Then,

- $\mathcal{G}(\mathcal{L}(P_1))$ and $\mathcal{G}(\mathcal{L}(P_2))$ are cycles of length 4
- If $\mathcal{L}(P_1) \cap \mathcal{L}(P_2) \neq \emptyset$, then either $\mathcal{L}(P_1) \cap \mathcal{L}(P_2) = \{L_1\}$ or $\mathcal{L}(P_1) \cap \mathcal{L}(P_2) = \{L_1, L_2\}$ where $\{L_1, L_2\}$ is an edge in $\mathcal{G}(\mathcal{L}(P_1))$ and $\mathcal{G}(\mathcal{L}(P_2))$.

With the first property of Lemma 5, for every $P_i \in I^*$, the transposition graph $G(\mathcal{L}(P_i), E_i)$ is a cycle of length 4, which we can also treat and call as a unit square. We can then denote $\mathcal{G}(\Upsilon) = G(\Upsilon, E)$ as $\cup_{C_i \in \mathcal{C}} C_i$, where $C_i = G(\mathcal{L}(P_i), E_i)$, $\Upsilon = \cup_{P_i \in I^*} \mathcal{L}(P_i)$, and $E = \cup_{i=1,\ldots,|I^*|} E_i$. In other words, we can say that $\mathcal{G}(\Upsilon)$ is composed of $|I^*|$ unit squares. From now on, we can also interchangeably call a unit square in $\mathcal{G}(\Upsilon)$ as a poset.

Now, from the second property, we can say that when two squares intersect – they either intersect in a vertex or in an edge in the graph. Since we are only dealing with the case where $\mathcal{G}(\Upsilon)$ is rectangular, then we know that the former case always holds. Moreover, we can say that for every edge $e \in E$, there is at least one and at most two unit squares $C_i = G(\mathcal{L}(P_i), E_i), C_j = G(\mathcal{L}(P_j), E_j)$, $C_i \neq C_j$ where $e \in E_i$ and $e \in E_j$. In other words, an edge in $\mathcal{G}(\Upsilon)$ is either covered by one unit square or two unit squares or, equivalently, by one or two posets in $I^*$.

As an example, if we build $\mathcal{G}(\Upsilon)$ where $\Upsilon$ is the set of linear orders in Figure 2a, then we have the transposition graph in Figure 3. The graph is rectangular and consists of three squares – namely $C_1$, $C_2$, and $C_3$ – which corresponds to $P_1, P_2$, and $P_3$, respectively in Figure 2b. Moreover, $C_1$ and $C_2$ intersect in an edge, as do $C_2$ and $C_3$. From this, we can then also treat the problem as finding the minimum set of unit squares that cover all the vertices of the transposition graph.

The pseudocode of Algorithm 1 for Rectangular Hammock(2,2)-Poset Cover Problem is shown below. The algorithm works by iterating through the set of posets in $I^*$ or equivalently the set of unit squares in $\mathcal{G}(\Upsilon)$. In every iteration, it selects one poset $P$ to be part of the optimal solution $P^*$. To determine the poset to be selected, it finds a linear order (vertex) that can be covered by exactly one poset (unit square). That single poset should then be selected and be part of the optimal solution. After selecting a poset, it then determines posets that can be trimmed. A poset can be trimmed if its remaining uncovered linear extensions can already be covered by exactly one other poset in the remaining set of candidate posets. The above steps are repeated until all the linear orders are already covered.

To illustrate the algorithm, let us consider again the simple posets in Figure 2 and 3. In the first iteration, $L_1$, $L_2$, $L_7$, and $L_8$ are the linear orders that are covered only by a single poset. If, for example, we have evaluated $L_1$ first, then we should add $P_1$ to $P^*$. After selecting $P_1$, its linear extensions $L_1, L_2, L_3$ and $L_4$ are now covered. The remaining uncovered vertices are given by $\Upsilon_{uncov} = \{L_5, L_6, L_7, L_8\}$. Now, we determine if there are posets in the remaining ones which is contained in $P^*_{cand} = \{P_2, P_3\}$, that can be trimmed. $P_3$ cannot be trimmed since its remaining uncovered

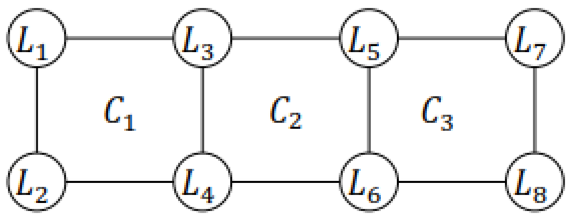


**Figure 3.** Transposition Graph G(ϒ).

linear extensions are covered by itself and $P_2$. On the other hand, $P_2$ can already be trimmed since its remaining uncovered linear extensions, which are $L_5$ and $L_6$, can already be covered by another exactly one poset, which is $P_3$.

After the first iteration, we can now visualize our current transposition graph like the one in Figure 4. When $P_2$ is trimmed, we can now say that $L_5$ and $L_6$ remains uncovered but are now covered only by a single poset in $P^*_{cand}$. Hence, in the next iteration, after evaluating $L_5, L_6, L_7$, or $L_8$, $P_3$ can already be selected to be part of the optimal solution. After selecting $P_3$, all the linear orders are now covered; hence, the Algorithm terminates and returns $\{P_1, P_3\}$ as the optimal solution.

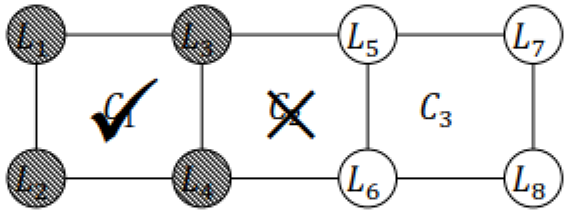


**Figure 4.** Example of an iteration in Algorithm 1.

**Algorithm 1**: Algorithm for Rectangular Hammock(2,2)-Poset Cover Problem

```
Input: A set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of linear orders on $V = \{1,2,\ldots,n\}$ where $\mathcal{G}(\Upsilon)$ is rectangular.
Output: A set $P^* = \{P_1, P_2, \ldots, P_k\}$ of hammock(2,2) posets where $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$ and $k$ is minimum.}
1  $I^* := GetAllHammock2kPosets(\Upsilon, 2)$
2  If $I^* = null$
3      return null
4  Else
5      Let $I^* = \{P_1, P_2, \ldots, P_p\}$
6      Set $P^*_{cand} := I^*$, $\Upsilon_{uncov} := \Upsilon$, $\Upsilon_{cov} = \emptyset$
7      for $i := 1$ to $m$ and $j := 1$ to $p$, set $M[i][j] = 1$ if $L_i \in \mathcal{L}(P_j)$, otherwise, $M[i][j] = 0$
8      while $\Upsilon_{uncov} \neq \emptyset$ do
9          Select $L_r \in \Upsilon_{uncov}$ where there is a unique $s \in \{1, \ldots, p\}$ such that $M[r][s] = 1$
10         $P^* := P^* \cup P_s$
11         $P^*_{cand} := P^*_{cand} \setminus P_s$
12         for every $L_t \in \mathcal{L}(P_s)$, set $M[t][s] = 0$, $\Upsilon_{uncov} := \Upsilon_{uncov} \setminus \{L_t\}$ and $\Upsilon_{cov} = \Upsilon_{cov} \cup \{L_t\}$
13         /* Trim removable posets */
14         for each $P_u \in P^*_{cand}$, sequentially do
15             $\mathcal{L}' := \mathcal{L}(P_u) \setminus \Upsilon_{cov}$
16             if there is a unique $P_v \in P^*_{cand}$, $P_u \neq P_v$ where $\mathcal{L}' \subset \mathcal{L}(P_v)$
17                 $P^*_{cand} := P^*_{cand} \setminus \{P_u\}$
18                 for every $L_w \in \mathcal{L}(P_u)$, set $M[w][u] = 0$
19     return $P^*$
```

***Theorem 6:*** Given a set of linear orders $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ over the set $V = \{1,2,\ldots,n\}$ where $\mathcal{G}(\Upsilon)$ is rectangular, Algorithm 1 returns a set of posets $P^* = \{P$ is hammock(2,2) poset$\}$ such that $\bigcup_{P_i \in P^*} \mathcal{L}(P_i) = \Upsilon$ and $|P^*|$ is minimum in $O(n^2 m^2 + m^3)$-time.

***Proof*:** From Theorem 1, we know that we can determine in $O(n^2 m^2)$-time the set $I^* = \{P | P$ is hammock(2,2) poset and $\mathcal{L}(P) \subseteq \Upsilon\}$ where $\bigcup_{P \in I^*} \mathcal{L}(P) = \Upsilon$. If $I^*$ is null, then it means that there exists no set of hammock(2,2) posets that covers $\Upsilon$. Hence, the algorithm also returns null in Line 3. On the other hand, in Line 5, we are already sure that $I^*$ contains all possible hammock(2,2) posets that cover $\Upsilon$. Let $I^* = \{P_1, P_2, \ldots, P_p\}$. Without loss of generality, let every

$P_i \in I^*$ be distinct, *i.e.* for every posets $P_1$ and $P_2$ in $I^*$, $<_{P_1} \neq <_{P_2}$. What remains then is to determine a subset of $I^*$ with minimum cardinality that covers $\Upsilon$.

Let $P^*$ be the intended output – the minimum set of poset that covers $\Upsilon$. Let $P^*_{cand}$ be the set of candidate posets that are eligible to be selected next for $P^*$. Let $\Upsilon_{uncov}$ be the set of linear orders that are not yet covered. Lastly, let $\Upsilon_{cov}$ be the set of linear orders that are already covered.

Now, we want to show that the following loop invariants hold in every iteration $i$ of the while loop body (lines 8–18).

1) At Line 9, there always exists $L_r \in \Upsilon_{uncov}$, that is covered by exactly one poset in $P^*_{cand}$, *i.e.* there is a unique $s \in \{1, \dots, p\}$ where $M[r][s] = 1$.
2) At Line 18, $P^*$ is the minimum set of Hammock(2,2)-Posets that covers $\Upsilon_{cov}$.

- When $i = 1$
  Initially, $\Upsilon_{uncov} = \Upsilon$. Since $\mathcal{G}(\Upsilon)$ is a rectangular graph and is composed of unit squares, then there exists a vertex $v$ that is incident to exactly 2 edges and these two edges are part of exactly one square. Equivalently, this means that there exists a linear order in $\Upsilon_{uncov}$ that is covered by exactly one poset, say $P_1$. Thus, invariant 1 is satisfied. Now, since only $P_1$ covers $L_r$, then it must be that $P_1 \in P^*$. Hence, $P^* = \{P_1\}$ is the minimum set that covers $\Upsilon_{cov} = \mathcal{L}(P_1)$. This satisfies invariant 2.
- Suppose the invariants hold for iteration $i = k$
- When $i = k + 1$
  In every iteration, the values in $M$ are changed into two parts of the algorithm. First, in Line 12, $M[u][v] = 0$ for $L_u \in \mathcal{L}(P_v)$ where $P_v$ is the poset chosen and transferred from $P^*_{cand}$ to $P^*$. Second is in Line 18, $M[u][v] = 0$ for $L_u \in \mathcal{L}(P_v)$ where $P_v$ is the poset trimmed. Note that in either update, the poset $P_v$ is removed from $P^*_{cand}$. Hence, we can say that $M[u][v] = 1$ if and only if $L_u \in \mathcal{L}(P_v)$ where $P_v \in P^*_{cand}$. We can then also say that $\mathcal{G}(\bigcup_{P \in P^*_{cand}} \mathcal{L}(P)) \subset \mathcal{G}(\Upsilon)$ is also a grid graph that consists of $|P^*_{cand}|$ unit squares, *i.e.* $\bigcup_{P_j \in P^*_{cand}} C_j$ where $C_j = G(\mathcal{L}(P_j), E_j)$. This implies that there exists a set of vertices, say $W$, from $\mathcal{G}(\bigcup_{P \in P^*_{cand}} \mathcal{L}(P))$ that is incident to exactly 2 edges and these two edges are part of exactly one square, or equivalently exactly one poset. In other words, $W = \{L_u | M[u,v] = 1 \text{ for exactly one } v\}$.

  Next, we show that there is at least one $L_r \in W$ that is not yet covered, *i.e.* $L_r \in \Upsilon_{uncov}$. To show this, suppose every $L_r \in W$ is already covered. Now, since $M[r][s] = 1$, we know that $P_s$ is the only poset in $P^*_{cand}$ that covers $L_r$. Let $C_s = G(\mathcal{L}(P_s), E_s)$ be the corresponding unit square of $P_s$ where $\mathcal{L}(P_s) = \{L_r, L_e, L_f, L_g\}$ and $E_s = \{\{L_r, L_e\}, \{L_r, L_f\}, \{L_e, L_g\}, \{L_f, L_g\}\}$. Since $L_r$ is covered, then it must have been covered by a poset in $P^*$, say $P_g$. As we mentioned earlier, $L_r$ is incident to two edges of $E_s$ that belong to $C_s$ only, *i.e.* there exists no other square $C_t = G(\mathcal{L}(P_t), E_t)$ in $\mathcal{G}(\bigcup_{P \in P^*_{cand}} \mathcal{L}(P))$ where $\{L_r, L_e\}, \{L_r, L_f\} \in E_t$. By Property 2 of the transposition graph, an edge is covered by at most 2 unit squares. Then, $P_g$ must have covered not only $L_r$ but a vertex or a linear order that is adjacent to $L_r$. Let it be the $L_e$ since $\{L_r, L_e\} \in E_s$. The remaining vertices are $L_f$ and $L_g$. Note that $\{L_f, L_g\}$ is also an edge in $E_s$. Hence, we have two cases. First, $\{L_f, L_g\}$ is a common edge of at most two unit squares or equivalently two posets in $I^*$. Then, $P_s$ should have been trimmed in the previous iteration. Second, $\{L_f, L_g\}$ is an edge of only one unit square or equivalently one poset, *i.e.* $P_s$. Then at least one of $L_f, L_g$ is in $W$ and not yet covered. Both cases arrive at a contradiction; hence, we can say that there exists at least one $L_r \in W$ that is also in $\Upsilon_{uncov}$ and invariant 1 holds.

  Now, let $L_r$ be the selected linear extension at the start of iteration $k + 1$ in Step 8 where there is a unique $s \in \{1, \dots, p\}$ where $M[r][s] = 1$. Moreover, let $P^*_i, P^*_{cand,i}, \Upsilon_{uncov,i}, \Upsilon_{cov,i}$ be the values of $P^*, P^*_{cand}, \Upsilon_{uncov}$, and

$\Upsilon_{cov}$ at the end of iteration $i$, respectively. To satisfy invariant 2, we must show that $P^*_{k+1} = P^*_k \cup \mathcal{L}(P_s)$ is the minimum set of posets that cover $\Upsilon_{cov,k+1} = \Upsilon_{cov,k} \cup \mathcal{L}(P_s)$.

Suppose, $P^*_{k+1}$ is not minimum, *i.e.* there exists $P^{*'}_{k+1}$ where $|P^{*'}_{k+1}| < |P^*_{k+1}|$ that covers $\Upsilon_{cov,k+1}$. This means that $|P^{*'}_{k+1}| < |P^*_k| + 1$ or, in other words, $|P^{*'}_{k+1}| \leq |P^*_k|$. Let $P^{*'}_k$ be also the subset of $P^{*'}_{k+1}$ that covers $\Upsilon_{cov,k}$. With our assumption that $P^*_k$ is the minimum set that covers $\Upsilon_{cov,k}$, then it must be that $|P^{*'}_k| \geq |P^*_k|$. Let us now consider the following cases:

- Case 1: There is only one poset in $I^*$, which is $P_s$, that cover $L_r$
  In this case, since $L_r$ is not yet covered, then $P_s$ must be part of the solution. Hence, $P^{*'}_k \cup \{P_s\} \subseteq P^{*'}_{k+1}$. Thus, $|P^{*'}_{k+1}| \geq |P^{*'}_k| + 1$. However, $|P^{*'}_k| \geq |P^*_k|$. Hence, it cannot be that $|P^{*'}_{k+1}| \leq |P^*_k|$.
- Case 2: $L_r$ is covered by more than one poset in $I^*$
  Without loss of generality, let $L_r$ be covered by two posets in $I^*$, $P_s$ and $P_t$.
  Since there is a unique $s \in \{1, \ldots, p\}$ where $M[r][s] = 1$ at the current iteration and $L_r \in \Upsilon_{uncov}$ means it is not yet covered by any poset in $P^*_k$, then $M[r][t]$, was set to 0 and removed in $P^*_{cand}$ in Step 14-18 during some previous iteration. Note that we trimmed poset when the remaining uncovered linear extension of the poset can already be covered by another and exactly one poset. Let the remaining uncovered linear extensions be $\mathcal{L}(P_t)_u = \{L \in \mathcal{L}(P_t) | L \in \Upsilon_{uncov,k}\}$. Note that $L_r \in \mathcal{L}(P_t)_u$. The exactly one poset that covers $\mathcal{L}(P_t)_u$ must be $P_s$. In other words, $\mathcal{L}(P_t)_u \subset \mathcal{L}(P_s)$. Choosing $P_t$ then will only add $\mathcal{L}(P_t)_u$ to the covered linear orders, but choosing $P_s$ will cover $\mathcal{L}(P_t)_u \cup (\mathcal{L}(P_s) \setminus \mathcal{L}(P_t)_u)$. Hence, $P_s$ is the optimal choice to cover $L_r$. This means that $P_s \in P^{*'}_{k+1}$. Hence, $P^{*'}_k \cup \{P_s\} \subseteq P^{*'}_{k+1}$. Hence, $|P^{*'}_{k+1}| \geq |P^{*'}_k| + 1$. However, $|P^{*'}_k| \geq |P^*_k|$. Hence, it cannot be that $|P^{*'}_{k+1}| \leq |P^*_k|$.

The two cases lead to contradiction; hence, $P^*_{k+1} = P^*_k \cup \{P_s\}$ is the minimum set that covers $\Upsilon_{cov,k+1}$.

Since the while loop terminates when $\Upsilon_{uncov}$ is empty, or in other words $\Upsilon_{cov} = \Upsilon$, then the final $P^*$ is the minimum set that covers $\Upsilon$.

Now, we determine the running time complexity of the algorithm.

One dominating part is the subroutine $GetAllHammock2kPosets(\Upsilon, 2)$ which has a running time of $O(2\, n^2\, m^2)$ where $n = |V|$ and $m = |\Upsilon|$ by Theorem 1. Since $\mathcal{G}(\Upsilon)$ is rectangular, then the return of the subroutine has a maximum size of $|I^*| \leq \left\lceil \frac{m}{2} \right\rceil - 1$. This is when the rectangle has only one row or one column. This means that evaluating $P^*_{cand}$ takes $O(m)$-time, while evaluating every element of the matrix $M$ takes $O(m^2)$-time.

Another dominating part is the nested loop, which consists of the outer while-loop and the inner for-loops. The outer while-loop iterates until all linear orders are covered; hence, it runs in $O(m)$-time. The most dominating part inside the while loop is the last for-loop in line 14, which iterates through $P^*_{cand}$ twice in $O(m^2)$-time. Hence, the running time of the nested loop is $O(m^3)$.

Overall, the algorithm runs in $O(n^2\, m^2 + m^3)$-time. ■

We have tried to apply the Algorithm 1 for grid graphs, which are super graphs of rectangular graphs. Note that grid graphs also capture the properties of the transposition graph of hammock(2,2) posets in Lemma 5. It is also a union of unit squares. Moreover, it captures the property that two unit squares can intersect in a vertex. However, in a general grid graph, the loop invariants in the proof do not always hold. Hence, a possible next step in determining the hardness of Hammock(2,2)-Poset Cover Problem is to study its hardness when the transposition graph is a general grid graph. There are also hard problems such as the Hamilton Path that have been shown to be in $P$ with rectangular graphs but are already NP-Hard in a general grid graph (Itai *et al.* 1982).

## CONCLUSION

In this paper, we have provided some complexity results for the Hammock($\underbrace{2,2,\ldots,2}_{k}$)-Poset Cover Problem, a variation of the Poset Cover Problem with the same input – set $\{L_1, L_2, \ldots, L_m\}$ of linear orders over the set $\{1,2,\ldots,n\}$, but the solution is restricted to a set of simple hammock($\underbrace{2,2,\ldots,2}_{k}$) posets . It is known that the problem is NP-hard when $k \geq 3$ but in $P$ when $k = 1$. The computational complexity of the problem when $k = 2$ is not yet known. In this paper, we have determined the approximation complexity of the cases that are NP-Hard. We have shown that the Hammock($\underbrace{2,2,\ldots,2}_{k}$)-Poset Cover Problem is in $APX$ when $k \geq 3$. Specifically, it is $1 + \frac{1}{2^k}$ - approximable. On the other hand, we have continued to explore the computational complexity of the problem when $k = 2$ (Hammock(2,2)-Poset Cover Problem). We have shown that it is in $P$ when the transposition graph of the input set of linear orders is rectangular. Other simple structures that agree with the property of the transposition graph of hammock(2,2) posets are grid graphs, which are supergraphs of rectangular graphs. Hence, one direction in determining the hardness of Hammock(2,2)-Poset Cover Problem is to study the hardness of the problem when the transposition graph of the input linear orders is a grid graph.

## REFERENCES

AUSIELLO G, CRESCENZI P, GAMBOSI G, KANN V, MARCHETTI-SPACCAMELA A, PROTASI M. 1999. Complexity and Approximation. Berlin/Heidelberg: Springer-Verlag.

AUSIELLO G, PASCHOS V. 2005. Approximability preserving reduction. Cahier Du Lamsade 227: 12.

CRESCENZI P. 1997. A short guide to approximation preserving reductions. Computational Complexity. Proceedings of the Twelfth Annual IEEE Conference on (Formerly: Structure in Complexity Theory Conference). p. 262–273.

ESSA H, EL-LATIF Y, ALI S, KHAMIS S. 2016. A New Approximation Algorithm for k-Set Cover Problem. Arabian Journal for Science and Engineering 41(3): 935–940.

FERNANDEZ P, HEATH L, RAMAKRISHNAN N, TAN M, VERGARA JP. 2013. Mining Posets from Linear Orders. Discrete Mathematics, Algorithms and Applications 5(4).

HALLDORSSON M.1996. Approximating k-set cover and complementary graph coloring. International Conference on Integer Programming and Combinatorial Optimization. Springer. p. 118–131.

HEATH L, NEMA A. 2013. The Poset Cover Problem. Open Journal of Discrete Mathematics 3: 101–111.

ITAI A, PAPADIMITRIOU C, SZWARCFITER J. 1982. Hamilton paths in grid graphs. SIAM Journal on Computing 11(4): 676–686.

ORDANEL I, ADORNA H, CLEMENTE J. 2017. Approximation of two simple variations of the Poset Cover Problem. In: Workshop on Computation: Theory and Practice (WCTP 2017). p. 12–13.

ORDANEL I, ADORNA H. 2018. Optimal Deterministic Algorithm for Hammock(2,2)-Poset Cover Problem. Philipp J Sci 147(7): 733–748.

PEI J, WANG H, LIU J, WANG K, WANG J, YU P. 2006. Discovering frequent closed partial order from strings. IEEE Transactions on Knowledge and Data Engineering 18(11): 1467–1481.

PRUESSE G, RUSKEY F. 1994. Generating Linear Extension Fast. SIAM Journal of Computing 23(2): 373–386.